

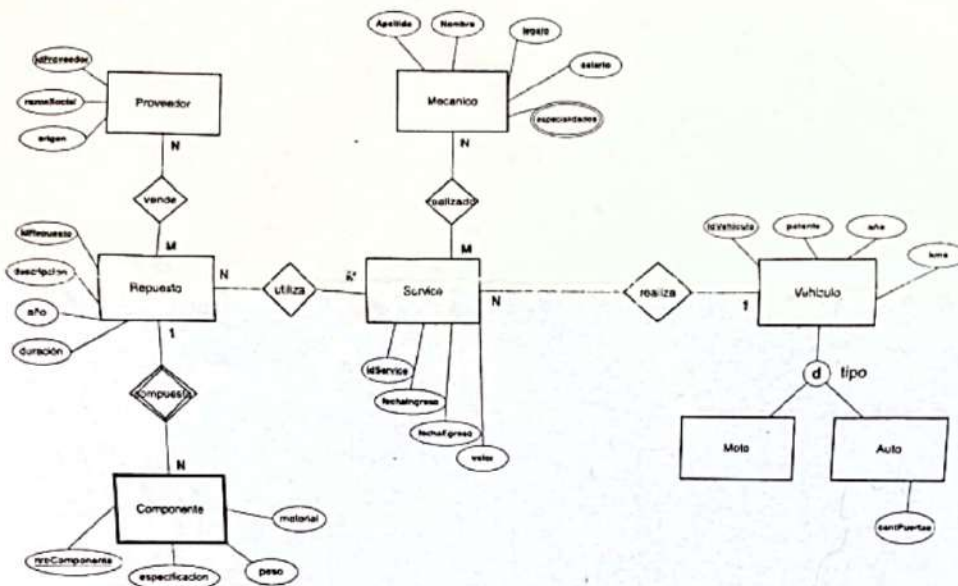
2 Parcial - Bases de Datos - 25/11/2022

- Debe identificarse **cada** hoja con nombre, apellido, LU y su **número de orden**.
- Complete la primera hoja con la cantidad total de hojas entregadas y numere todas las hojas.
- Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Para que un ejercicio sume puntos **no deben cometerse errores conceptuales graves**.
- La **interpretación** del enunciado forma parte de la evaluación.

Criterio de Aprobación: Se aprueba con 7. Ejercicio 1 5ptos, Ejercicio 2 5ptos, Debe sumar al menos 3 en cada ejercicio.

1. NOSQL

Dado el siguiente DER que modela los datos para una aplicación que administra trabajos en vehículos



Se pide:

(2.5) a) Document Database: **dibujar el diagrama de interrelación de documentos**, justificando las decisiones tomadas, sabiendo que:

- Dado un service, se requiere conocer el año y la duración de los repuestos utilizados en el mismo.
- Cada vez que se consulta por un mecánico se deben conocer todos los datos de los services en los que trabajó así también como la patente del vehículo.
- Dado un vehículo se quiere conocer de los services que le fueron realizados y sus valores correspondientes

Especificar en JSON Schema el tipo de documento Repuesto.

(1.2) b) Realizar el diseño Column Family haciendo el diagrama de Chebotko para las siguientes consultas:

- Dado una patente, mostrar año y km del vehículo y los services que se fueron realizando, ordenados por valor, en forma descendiente.
- Mostrar todos los repuestos y la razón social de los proveedores, ordenados por la duración del repuesto

(1.0) c) Diseña una base de datos clave valor para el siguiente comportamiento: Se requiere que un mecánico pueda ingresar al sistema (login) y ver sus datos. También se desea que pueda acceder a la lista de todos los services en los que trabajó, poder acceder a los detalles de los mismos y a todos los datos del vehículo que se trabajó en cada caso.

2. Concurrency and Recoverability

(4,5) a) Dada la siguiente historia

$H = w_1(A); r_2(A); r_3(A); w_3(D); r_1(B); r_3(C); r_3(B); w_2(C); c_3; w_5(B); w_4(E); c_2; r_4(D); w_5(E); c_4; w_5(C); c_5; c_1$

- (i) Hacer el grafo de precedencia de H, indicar si es serializable y en caso afirmativo indicar todas las historias seriales equivalentes.
- (ii) Clasificar H con respecto a recuperabilidad: ¿Es recuperable? ¿Evita aborts en cascada? ¿Es estricta? Justificar las respuestas

(2) b) Suponga la siguiente historia tentativa parcial sobre un planificador con timestamp monoversion:

$H_2 = r_1(X); c_1; r_3(Y); r_2(Y); w_3(X); r_2(Z); r_3(Z); w_2(Z); c_3; w_2(X); c_2$

- (i) Analice (JUSTIFIQUE) que pasaría en cada uno de los siguientes casos si los *timestamps* fueran los indicados:

(a) $T_1 = 100, T_2 = 200, T_3 = 300$

(b) $T_1 = 100, T_2 = 300, T_3 = 200$

- (ii) Para el punto anterior: ¿Cambiaría algo si el planificador fuera MVTO? JUSTIFIQUE

(1,5) c) Luego de la caída de una base de datos, el registro de recuperación de tipo *undo-log non-quiescent checkpoint* contiene los siguientes datos:

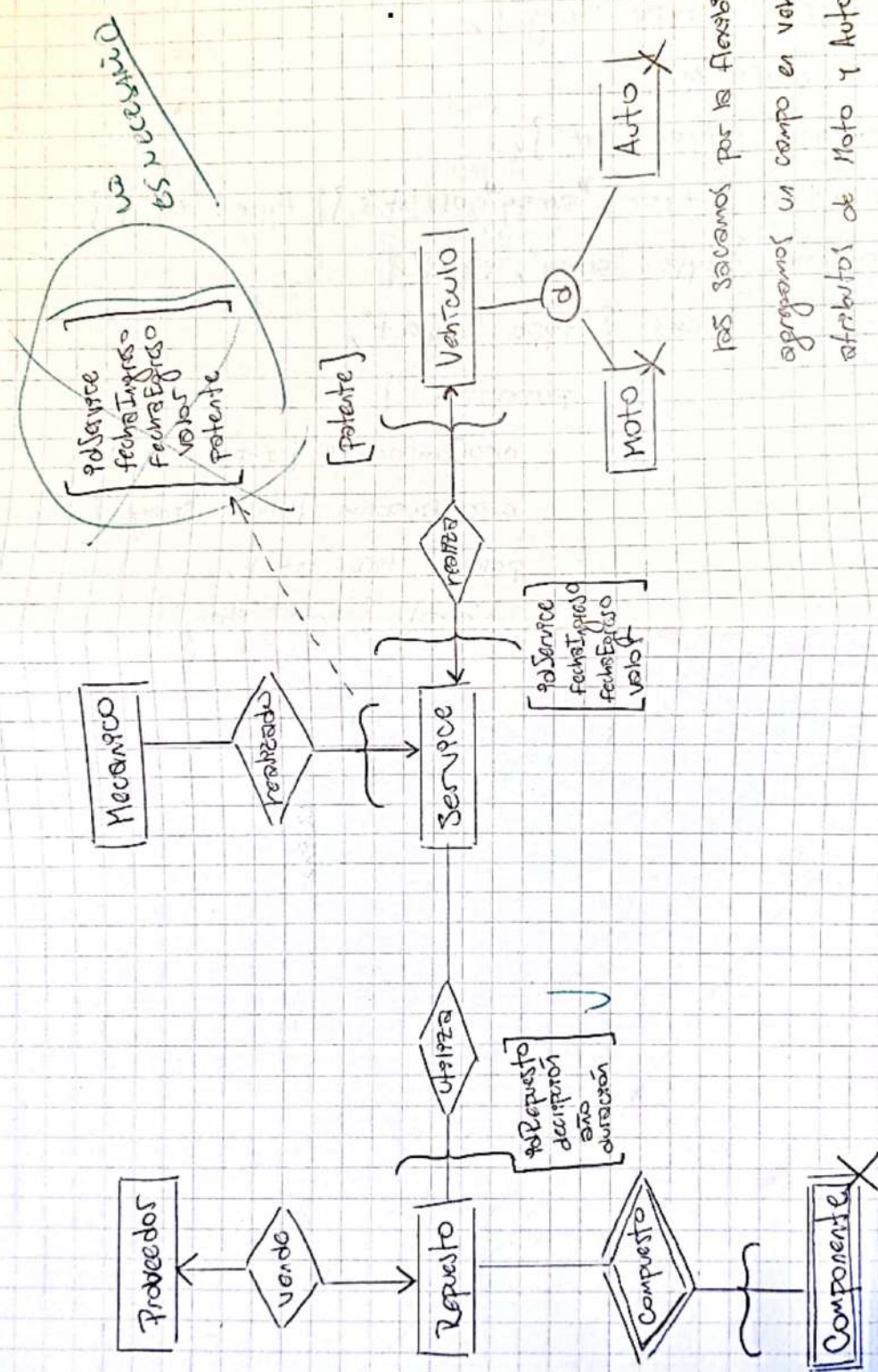
```

<START T1>
<T1, X1, 1>
<START CKPT ???>
<START T2>
<T2, X2, 2>
<T1, X1, 3>
<START T3>
<COMMIT T1>
<END CKPT>
<START CKPT ???>
<T2, X2, 4>
<T3, X3, 5>
<START T4>
<COMMIT T2>
<T4, X4, 6>
<COMMIT T3>
<END CKPT>
<START T5>
<T5, X5, 7>
<START CKPT ???>
<T4, X4, 8>
CRASH !!!

```

- (i) ¿Cuáles son los valores correctos de los tres registros <START CKPT ???>? Es decir, completar con lo que corresponda donde dice <??>.
- (ii) Suponiendo que los tres registros <START CKPT >están correctamente almacenados en el log, según su respuesta en el ítem anterior, muestre qué elementos son recuperados por el gestor de recuperación y calcular los valores tras la recuperación.
- (iii) Indicar qué fragmento del registro necesita leer el gestor de recuperación.
- (iv) ¿Qué modificaciones ocurren en el log si las hubiera?

Ejercicio 1



de los documentos y las sacamos por la flexibilidad y agregamos un campo en vehiculo y los atributos de Moto y Auto.

para saber si es auto o moto ✓



Json Schema de Repuesto:

```
{  
  type: "object",  
  properties: {  
    idRepuesto: { type: "int" },  
    descripcion: { type: "string" },  
    año: { type: "int" },  
    duracion: { type: "int" },  
    proveedor: { type: "array", items: { type: "int" } },  
    componentes: { type: "array",  
      items: { type: "object",  
        properties: {  
          nroComponente: { type: "int" },  
          especificacion: { type: "string" },  
          peso: { type: "int" },  
          material: { type: "string" }  
        }  
      }  
    }  
  }  
}
```

}

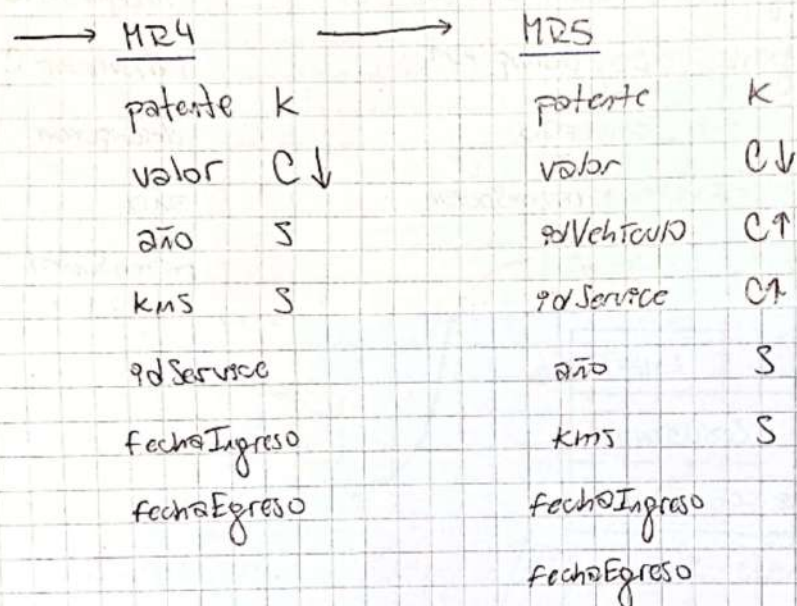
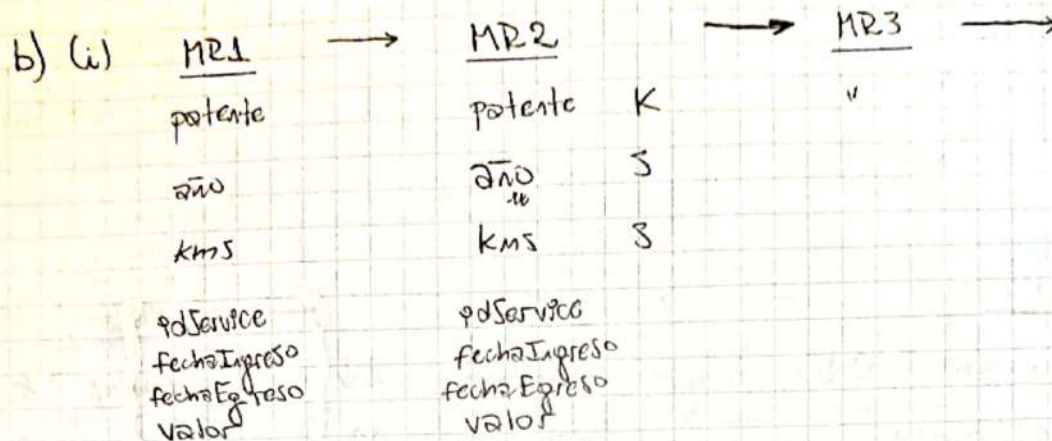
}

}

}

✓

Segue Ejercicio 1



(ii) MR1 → MR2 → MR3 →

1º Repuesto " "

descripcion " "

año " "

duracion " "

razon Social " "

→ MR4 → MR5 → MR6

duracion C↓ duracion C↓ dummy K

1º Repuesto 1º Repuesto C↑ duracion C↓

descripcion 1º Proveedor C↑ 1º Repuesto C↑

año descripcion 1º Proveedor C↑

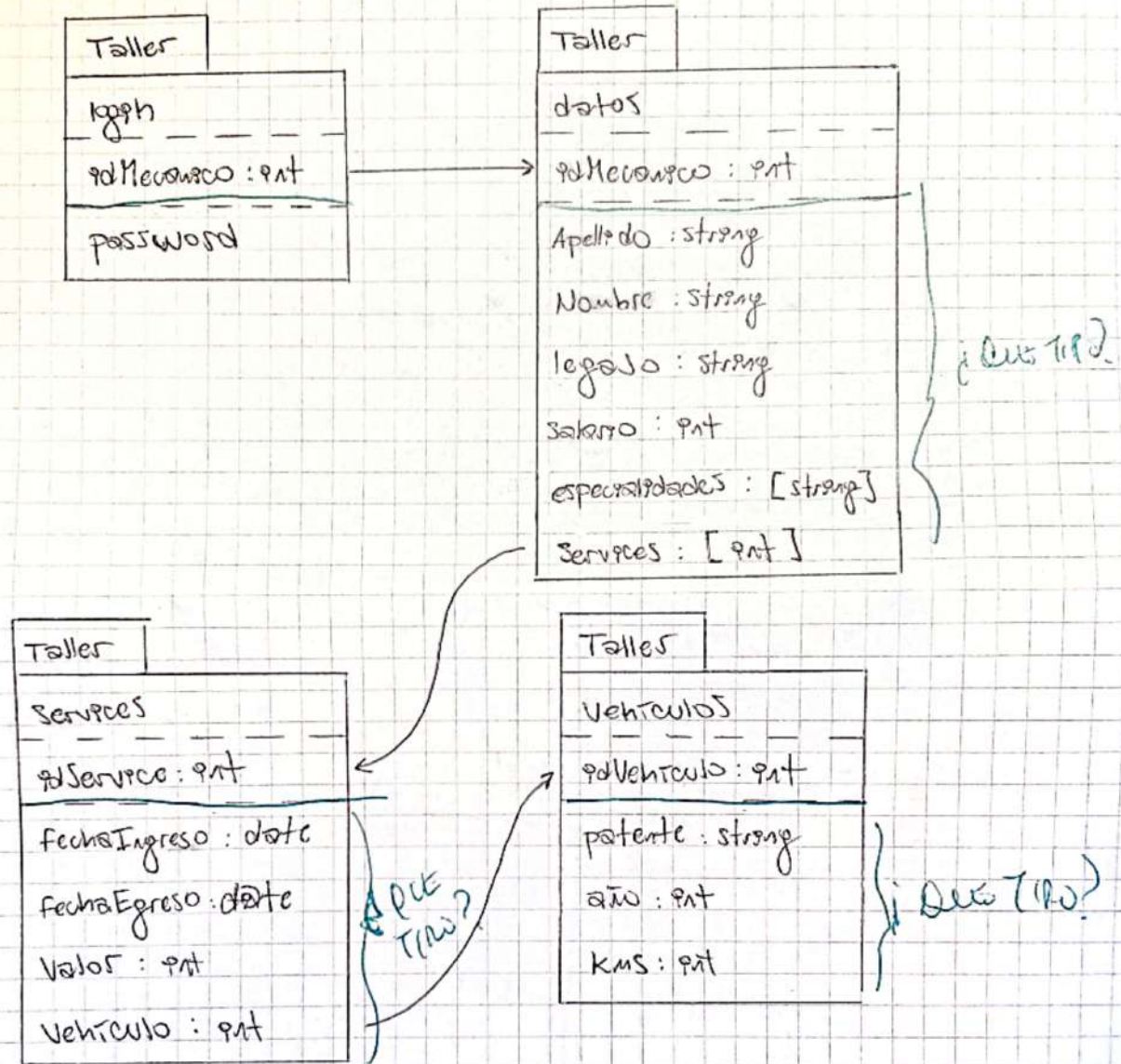
duracion año descripcion

razon Social razon Social año

razon Social razon Social razon Social ✓

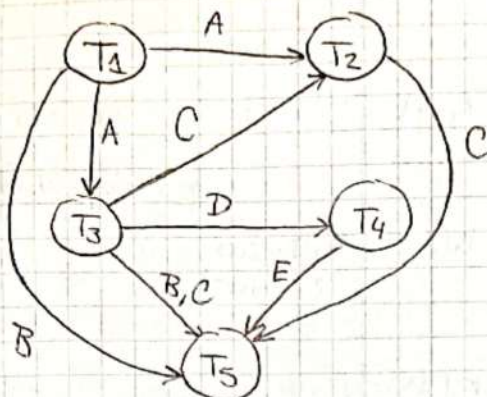
Segu Ejercicio 1

c) Agregamos idMecanico y password como atributos al DER a la entidad Mecanico.



Ejercicio 2

a) i)



Es serializable por ser acíclico. Historial serializable equivalente:

• T_1, T_3, T_2, T_4, T_5

FACTA T_1, T_3, T_4, T_2, T_5

ii) No es recuperable porque T_2 lee A de T_1 pero C_2 ocurre antes que C_1 .

Como no es recuperable tampoco es ACA ni ST. ✓

b) i) (a)	T_1 100	T_2 200	T_3 300	X RT=0 WT=0	Y RT=0 WT=0	Z RT=0 WT=0
$r_1(x)$				RT=100		
C_1						
			$r_3(y)$		RT=300	
		$r_2(y)$			Mantengo RT	
			$w_3(x)$	WT=300		
		$r_2(z)$				RT=200
			$r_3(z)$			RT=300
		$w_2(z)$				Write too late ✓
			C_3	C=True		Aborto T_2 ! ✓
		$w_1(x)$				
C_2						

(b)

T_1 100	T_2 300	T_3 200	X RT=0 WT=0	Y RT=0 WT=0	Z RT=0 WT=0
$r_1(x)$ c_1			RT=100		
		$r_3(y)$		RT=200	
	$r_2(y)$			RT=300	
		$w_3(x)$	WT=200 C=False		
	$r_2(z)$				RT=300
		$r_3(z)$			
	$w_2(z)$			WT=300	
		c_3	C=True		
	$w_2(x)$		WT=300 C=False		
	c_2		C=True		

(ii) ~~No~~ No cambia en el primer caso porque multiversionado nos salva de los read-too-late y no de los write-too-late, como sucedió. ✓

c) (i) $\langle \text{START CKPT } T_1 \rangle$

$\langle \text{START CKPT } T_2, T_3 \rangle$

$\langle \text{START CKPT } T_4, T_5 \rangle$ ✓

(ii) Como es UNDO-log, tenemos que ~~hacer~~ hacer UNDO de las transacciones no comprometidas o abortadas empezando a mirar desde el último ~~START~~ ~~END~~ CKPT* y agregando los que empezaron antes y no terminaron para el START CKPT, es decir, los que escribimos en (i). ✓

Vemos que solamente T_4 y T_5 no terminaron, entonces desde el final al principio aplicamos ~~B~~ UNDO: ✓

$X_4 \leftarrow 8$

$X_5 \leftarrow 7$

$X_4 \leftarrow 6$

(iii) Respondido en (ii).

(iv) Agregamos ~~abort~~ ^{en el log} a las transacciones que hacemos UNDO.

$\text{ABORT}(T_4)$ y $\text{ABORT}(T_5)$

* tal que exista un END CKPT. ✓